9-1994

# Parametric Volume Models for Interactive Three-Dimensional Grid Generation

Jawad D. Mokhtar
*Iowa State University*

James H. Oliver
*Iowa State University*, oliver@iastate.edu

DE-Vol. 69-1, Advances in Design Automation – 1994
Volume 1
ASME 1994

# PARAMETRIC VOLUME MODELS FOR INTERACTIVE THREE-DIMENSIONAL GRID GENERATION

**Jawad D. Mokhtar and James H. Oliver**
Department of Mechanical Engineering
Iowa Center for Emerging Manufacturing Technology
Iowa State University
Ames, Iowa

## ABSTRACT

A method based on non-uniform rational B-spline (*NURBS*) curve, surface, and solid technology is presented for interactive grid generation of three-dimensional flow fields encountered in turbomachinery applications. The method allows construction of several types of multi-block grids including *H*-, *O*-, and *C*-grids for two-dimensional grids, and strict *H*-grids for three-dimensional grid generation. Automated two-dimensional block construction is facilitated via a traversal method that searches four-sided regions in the initial block structure. A *NURBS* surface is then constructed on each block by transfinite interpolation of the boundary curves themselves, and various point distribution options may be applied. Three-dimensional grid generation is an extension of the two-dimensional procedure. Tri-parametric hyperpatches (*NURBS* solids) are constructed from the two-dimensional block surfaces and grids may be generated via several point distribution functions. This method exploits existing geometric design data via Initial Graphics Exchange Specification (IGES) input of *NURBS*-based component geometry. It provides an efficient and robust method for complex grid generation to support a variety of analysis functions. Several example applications are presented to demonstrate the capabilities of the technique.

## INTRODUCTION

The modern turbomachinery design process typically involves computation of flow characteristics through multiple blade-row machinery via Navier-Stokes flow solvers. (Dawes, 1990; Denton, 1990, Chima, 1990) These analysis techniques are based on finite difference methods, and thus require carefully crafted point meshes as input. Interactive grid generation has become popular for a variety of applications in aerospace engineering, including turbomachinery design (Soni and Shih, 1990; Beach, 1989). Since the flow fields associated with turbomachinery can be comprised of very complex geometry, the primary advantage of an interactive grid generator is that it provides for substantial productivity gains in the analysis and design process. Although most turbomachinery manufacturers employ some form of this technology in their development process, automation tools are generally developed "in-house" and remain proprietary.

Several public domain tools for interactive grid generation, based on research conducted at NASA Lewis Research Center (Miller, 1994; Crook and Delaney, 1991), have recently emerged. These grid generation techniques (and the techniques referenced above) follow a similar methodology. First, a two-dimensional region is partitioned with curves to characterize the overall shape and topology of the flow field. The partition may incorporate boundaries of physical components as well as arbitrary non-physical divisions of the flow field. The curves are intersected and blocks are interactively constructed from topologically rectangular collections of curve segments. Several different curve representations have been considered such as damped cubic splines (Lee and Loellbach, 1989); piecewise Bezier segments (Beach, 1989; Soni and Shih, 1990). A variety of parametric biasing techniques are used to evaluate the curves (regardless of the curve representation) to distribute points on the boundaries of the blocks. In most methods the underlying parametric curve boundaries are essentially discarded, after this step. For two-dimensional grids, points are generated on the interior of the block via transfinite interpolation (Farin, 1993) based solely on the boundary points, not the curves from which the points were generated. For three-dimensional grids, this approach is generalized by applying transfinite interpolation to volume bounding surface grid points.

These approaches suffer from two fundamental aspects which limit their effectiveness. First, these methods generally do not exploit the geometric models of components typically available via contemporary CAD/CAM systems, because the boundary curves

are based on non-standard geometric bases. As a result, component geometry input is tedious and the resulting definition is approximate because parts are effectively re-modeled using the geometric utilities of the grid generation package. Second, the accuracy and flexibility of these methods is limited by the restricted application of parametric splines. In the geometric modeling literature, tensor product spline representations are commonly used for smooth characterization of sculpted geometry via two (surface) and three (volume) independent variables. However, with these methods, interior point generation neglects the true nature of the flow boundaries because the parametric spline curve definitions are used only for initial point distribution. In other words, surface and volume point distributions are generated from discrete approximations of the boundaries, so the approximation error is propagated throughout the grid. Furthermore, changes in desired grid density require complete repetition of the surface (and volume, for three-dimensional grids) transfinite interpolation functions. Thus, these methods can be quite computationally demanding.

This paper presents a technique for interactive three-dimensional grid generation for turbomachinery using non-uniform rational B-spline (*NURBS*) curve, surface, and solid technology. The method allows construction of various types of multi-block grids for two-dimensional grids including: *H*-, *O*-, and *C*-grids, and supports strict *H* -grids for three-dimensional grid generation. Typical *NURBS*-based CAD-generated turbomachinery component geometry may be read via the IGES (NIST, 1992) file format. Alternatively, the user may interactively fit input point profiles with integrated *NURBS* modeling utilities. These same utilities are available to interactively augment the flow field region with non-physical *NURBS* curves for initial block definition. Automated two-dimensional block construction is facilitated via a traversal method that searches four-sided regions in the initial block structure. A *NURBS* surface is then constructed on each block by transfinite interpolation of the boundary curves themselves, and various point distribution options may then be applied. Three-dimensional grid generation is an extension of the two-dimensional procedure. Tri-parametric hyperpatches (*NURBS* solids) are constructed from the two-dimensional block surfaces, and grids may be generated via several point distribution functions. Several example applications are presented to demonstrate the capabilities of the technique.

## GEOMETRIC UTILITIES

General descriptions of B-spline curves and their various forms can be found in most geometric modeling textbooks (e.g., Farin, 1993; Mortenson, 1985) or any of several survey articles on the topic (Böhm et al., 1984; Piegl,1991). A brief description of B-splines is provided to facilitate the subsequent development of this grid generation technique.

The B-spline basis functions are typically generated via the Cox-deBoor algorithm (Cox, 1972; deBoor, 1972). Given knot vector $\boldsymbol{T} = \{t_0, \ldots, t_i, t_{i+1}, \ldots, t_m\}$ a monotonically increasing sequence of real numbers, the $i^{th}$ B-spline basis function of degree, $p$ (order, $p+1$), denoted $N_{i,p}(t)$, is defined by the recursive relationship,

$$N_{i,0}(t) = \begin{cases} 1 & if \ (t_i \le t < t_{i+1}) \\ 0 & otherwise \end{cases}$$

$$N_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} N_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(t) \quad (1)$$

where it is understood that $0 / 0 = 0$. A B-spline surface of degree $(p, q)$ is specified by an $(m\text{-}p) \times (n\text{-}q)$ grid of control points $P_{ij}$ arranged in a topologically rectangular array and knot vectors $U$ and $V$ of length $(m + 1)$ and $(n + 1)$, respectively. The surface, denoted as $S(u,v)$, is thus defined as the tensor product of the control point array and the B-spline basis functions defined over each knot vector:

$$S(u,v) = \sum_{i=0}^{(m-p-1)} \sum_{j=0}^{(n-q-1)} N_{i,p}(u) N_{j,q}(v) P_{ij} \quad (2)$$

The knot vector governs the relationship between parametric and spatial variation, and its entries represent the parameter values at segment joints (knots). A non-periodic B-spline is characterized by a knot vector in which the first and last knot values are repeated $p + 1$ (order) times. This results in a surface that interpolates the control points on the edge of the rectangular array. A B-spline is characterized as uniform if the difference between successive interior knots is constant. A rational B-spline allows assignment of a scalar weight factor to each control point. Homogeneous coordinates are used to represent a rational surface in $r$-dimensional space as a polynomial in $(r+1)$-dimensional space. A projective mapping is used to recover the resulting $r$-dimensional surface.

Using this terminology the B-spline surface can be extended to a tri-parametric solid (hyperpatch) as follows,

$$T(u,v,w) = \sum_{i=0}^{(m-p-1)} \sum_{j=0}^{(n-q-1)} \kappa(i,j,p,q)$$

$$\kappa(i,j,p,q) = \sum_{k=0}^{(l-r-1)} N_{i,p}(u) N_{j,q}(v) N_{k,r}(u) P_{ijk} \quad (3)$$

Despite their thorough documentation in the open literature, computational implementation of general purpose *NURBS* modeling techniques is a large and complex undertaking. Thus, this research takes full advantage of the *DT_NURBS* Spline Library provided by the Naval Surface Warfare Center (US Navy, 1993). This library provides a complete and robust collection of *NURBS*-based modeling and analysis functions and is available for public distribution in the United States.

A transfinite interpolation method (Farin, 1993) is used to develop *NURBS*-based spline curves and solids for both two- and three-dimensional grid development. As shown in Figure 1 the method generates a surface given four boundary curves.

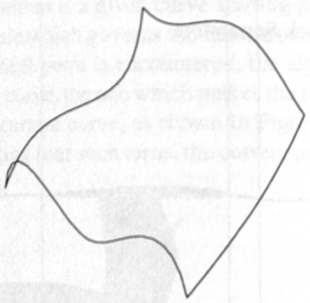

Figure 1 Boundary curves

Consider four NURBS boundary curves of the form

$$A_i(u) = \sum_{j=0}^{n} N_{j,p}(u) P_{i,j} \qquad i = 0, 1 \qquad u \in [0, 1]$$
$$B_i(v) = \sum_{k=0}^{m} N_{k,q}(v) Q_{i,k} \qquad i = 0, 1 \qquad v \in [0, 1] \tag{4}$$

where it is assumed that corner points are coincident, i.e., $A_0(1) = B_0(0), B_0(1) = A_1(1), A_1(0) = B_1(1), B_1(0) = A_0(0)$. Given the affine invariance property of *NURBS* curves, a tensor product surface which interpolates the boundary curves may be defined by transfinite interpolation of their control points. To determine the interior control points of the surface, a Boolean sum of the boundary control points is performed,

$$R_{i,j} = f(P)_{i,j} + f(Q)_{i,j} - f(PQ)_{i,j} \tag{5}$$

where the $f(P)$ and $f(Q)$ are defined by ruled surfaces between adjacent boundaries,

$$f(P)_{i,j} = f_0(t_j) P_{0,i} + f_1(t_j) P_{1,i}$$
$$f(Q)_{i,j} = f_0(t_i) Q_{0,j} + f_1(t_i) Q_{1,j} \tag{6}$$
$$t_i = 0, \frac{1}{n}, \frac{2}{n}, \ldots, 1 \qquad t_j = 0, \frac{1}{m}, \frac{2}{m}, \ldots, 1$$
$$i = 0, 1, \ldots, n \qquad j = 0, 1, \ldots, m$$

$f(P,Q)$ is a ruled surface defined by the corner points,

$$f(PQ) = f_0(t_i)\,[f_0(t_j) Q_{0,0} - f_1(t_j) Q_{0,m}] + f_1(t_i)\,[f_0(t_j) Q_{1,0} - f_1(t_j) Q_{1,m}] \tag{7}$$

and $f_0$ and $f_1$ are the cubic Hermite polynomials,

$$f_0(t) = 2t^3 - 3t^2 + 1$$
$$f_1(t) = -2t^3 + 3t^2 \tag{8}$$

Thus the *NURBS* surface is defined as:

$$S(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} N_{i,p}(u) N_{j,q}(v) R_{i,j} \tag{9}$$

Figure 2 shows a example of this transfinite interpolation method. The method is easily extended to form general tri-parametric solids (hyperpatches) from six boundary surfaces.

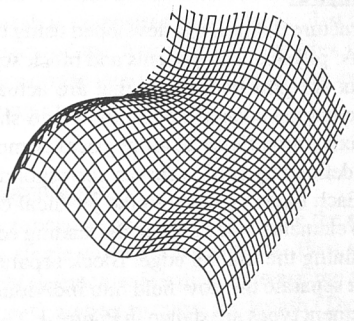

Figure 2 *NURBS* surface from transfinite interpolation of boundaries

## TWO-DIMENSIONAL GRID GENERATION

Two-dimensional grid generation begins with input component geometry in *NURBS* form via an *IGES* file, or by fitting input point profiles with *NURBS* interpolation utilities. Most input geometry for two-dimensional grid generation is planar. Axisymmetric components such as the hub and shrouds are represented as curves in the $r$-$z$-plane, assuming symmetry about the $z$-axis. The exception is blade models which are generally complex three-dimensional surfaces. In this work, blade surfaces are assumed to be modeled (Oliver et. al, 1994) with the $u$-parameter corresponding to the chordwise direction and the $v$-parameter corresponding to the span-wise direction, so the leading edge lies on a surface iso-parameter curve at $u = 0.5$, and the trailing edge at $u = 0$ and $u = 1$. These iso-parameter curves are projected onto the $r$-$z$-plane by setting the θ-coordinates of their control points to zero.

The flow field for structured multi-block grids is separated into several rectangular regions called blocks. Multi-block grids are generally characterized by their topological structure. The three primary types are known as: $H$-, $O$- and $C$-grids, as shown in Figure 3. The following discussion addresses grid topological layout, topological modification, block generation, and grid generation.

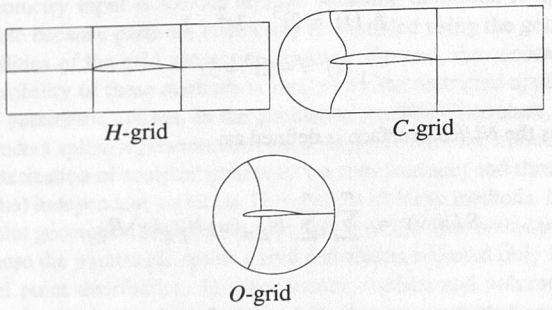


Figure 3 Grid types

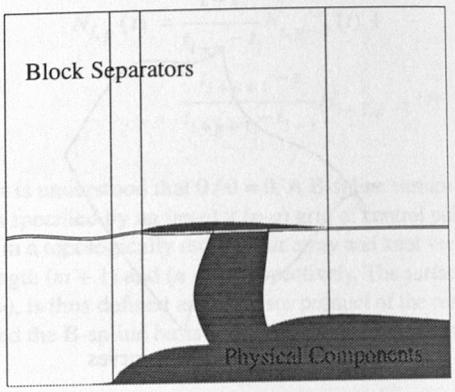


Figure 4 Grid element types

## Topological layout

The block structure definition is developed using two types of geometry elements: physical components and block separators. Physical components are grid boundaries that are actually part of the geometry of the engine such as cowls, mid-span shroud, hub, and blades. In the axisymmetric case, all physical components, except the blades, are defined by profile curves corresponding to surfaces of revolution. Each blade is added to the physical component data structure as two elements, one defining the leading edge of the blade and another defining the trailing edge. Block separators are imaginary curves that separate the flow field into individual blocks. Both of these grid element types are shown in Figure 4. The identical data structure is used to represent both the physical components and block separators; it is defined as follows: (All data structures in this paper are represented in the *C* programming language style).

```
struct Flow
{
        int type;
        double *curve;
        int nAttach;
        Attach **atchmnt;
};
```

The variable `type` indicates the curve type: 1, for open geometry; 2, for close geometry; 3, for leading edge; 4, for trailing edge; and 5, for block separator. The remaining variables are defined as follows: `curve` is the *NURBS* curve definition; `nAttach` is the number of other elements which are in contact with the curve; and `atchmnt` defines each of the contact points. The variable `atchmnt` takes the following data structure:

```
struct Attach
{
        double param;
        int type;
        int num;
        double paramTo;
        Attach *next;
};
```

The variables in this structure are defined as follows: `param` is the parameter value on the original curve; `type` is the type of element which is attached to the original curve (as described above); `num` is the element number associated with the element type; `paramTo` is the parameter value on the attached curve at which the attachment occurs; `next` is a pointer to another attachment if more than one attachment occurs at a given parameter value.

The attachment list for each element is determined by the intersections among all geometry elements. Essentially, each element is checked against all other elements and if an intersection exists it is added to the attachment list for that element. Once all elements have been checked against each other, the attachment list for each element is then sorted based on parameter value and multiple attachment points are combined in the next data structure.

## Topological Modification

The topological structure of the multi-block layout may be modified by the addition, deletion, and shape modification of the individual elements. Addition and deletion are affected by insertion and removal of geometry or flow separators from the data structure, respectively. Shape modification applies only to block separators, since physical components cannot be altered. Local geometric shape modification is achieved by applying translations to the control points or by inserting knots (Böhm, 1985) and hence control points, for additional shape control. Also knot removal is provided to reduce the complexity of the NURBS curves (Tiller, 1992). After any of these modifications, the intersection process is repeated to update the block data structure.

## Block Generation

Block generation is accomplished via construction of two-dimensional *NURBS* surfaces which define the multi-block grid. The surface is constructed using the transfinite interpolation method which requires that four boundary curves of the block be defined. The following data structure defines a block element:

```
struct Block
{
        Side *side[4];
        double *surf;
}
```

The variable `side[4]` defines each of the four boundary curves of the grid block, and `surf` is the two-dimensional *NURBS* surface that defines the block.

The multi-block data structure is comprised of intersecting curves representing component geometry and flow separators. Since each block is defined as a collection of segments of these curves, a method to traverse the data structure is developed to de-

termine the four boundary curve segments for each block. The basis of the traversal algorithm is a given curve starting point and direction, and a simple rule which governs the data structure navigation; i.e., as each attachment point is encountered, the algorithm selects as the next traversal curve, the one which makes the minimum angle with respect to the current curve, as shown in Figure 5. A block is formed if, after making four such turns, the current point is identical to the starting point.

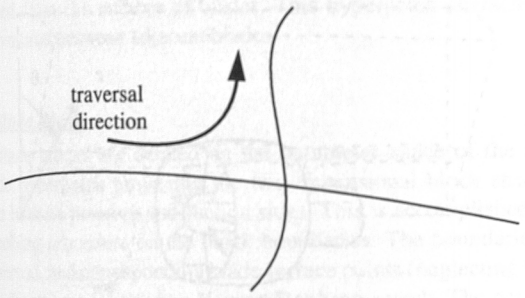


Figure 5 Minimum angle turn

This traversal process creates all the blocks in the grid, but leaves them in random position and orientations. Since most flow solvers require the grid to be output in a regular fashion, a method to order and orientate the blocks is required. The approach taken here is an interactive process that requires the user to select the first block along with its flow direction. The algorithm then scans the multi-block data structure to determine the adjacent blocks in the flow direction of the grid structure by comparing the edge of the outbound flow to all the edges of the remaining non-orientated blocks. After an edge is matched its corresponding block is oriented to the correct position with respect to the input flow direction, and it is added to a block group list. This procedure continues with the new block until no other blocks are found with edges adjacent to the out-bound flow of the current block. This method orders and orients only a single row at a time. Thus, the user is required to select the first block of every row.

An example showing the complete block traversal procedure is shown in Figure 6. As each block is formed, the four curves required to construct a block surface are defined with each side having the following data structure:

```
struct Side
{
        int type
        int num
        double *flow
        double *trim
        double pStart;
        double pEnd;
        ...
        ...
}
```

The variables `type` and `num` define the curve; `flow` is a pointer to the curve structure; `pStart` and `pEnd` are the starting and ending parameter values of the block's side, respectively along the parent curve; and `trim` is the normalized trimmed curve segment defined by `pStart` and `pEnd`.

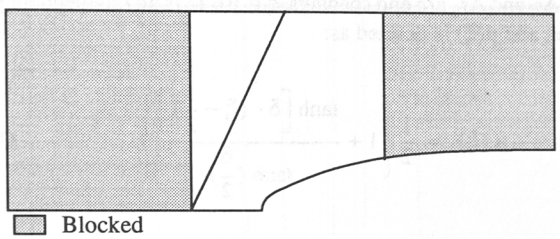


Figure 6 Block traversal example

## Grid Generation

The actual grid generation process requires the creation of a regular mesh of points on each of the block surfaces. The method used to generate the mesh is comprised of two steps: first the edge point distribution on each of the four block boundaries is determined; and, second, points on the interior of the surface are evaluated based on the edge spacing specification.

### Edge point distribution

The following procedure is used to determine the edge point distribution along a single side of a given block. First a geometric or hyperbolic stretching function is called to determine a spatial distribution of points with respect to curve arc length (Thompson, et al., 1985). Then the corresponding curve parameter value for each edge point is calculated as an inverse problem using a Newton/Raphson search.

The geometric stretching function is determined by dividing the curve arc length into two parts (referred to as the upper and lower portions) and expanding each into a polynomial series:

$$S = \Delta s_u \sum_{n=0}^{n_u - 1} r_u^n + \Delta s_l \sum_{n=0}^{n_l - 1} r_l^n \tag{10}$$

where $S$ is the total arc length of the curve, $r_u$ and $r_l$ are the stretching factors for upper and lower portions, $\Delta s_u$ and $\Delta s_l$ are the corresponding end spacings, and $n_u$ and $n_l$ are the desired number of points on each portion. Since the upper and lower portions must have equal spacing at the joint, the following relationship must hold,

$$\Delta s_l r_l^{nl-1} = \Delta s_u r_u^{nu-1} \tag{11}$$

Thus, there are two equations and four unknowns. The user may supply either the stretching factors, $r_u$ and $r_l$ or the end spacing $\Delta s_u$ and $\Delta s_l$.

The hyperbolic stretching function is determined from the following relationship:

$$s(\xi) = \frac{S \cdot \mu(\xi)}{\sqrt{\Delta s_2 / \Delta s_1} + (1 - \sqrt{\Delta s_2 / \Delta s_1})\,\mu(\xi)} \tag{12}$$

where $\Delta s_l$ and $\Delta s_u$ are end spacings, $\xi \in [0, 1]$ is an independent parameter, and $\mu(\xi)$ is defined as:

$$\mu(\xi) = \frac{1}{2}\left\{1 + \frac{\tanh\left[\delta \cdot (\xi - \frac{1}{2})\right]}{\tanh(\frac{\delta}{2})}\right\} \quad (13)$$

where $\delta$ is solved numerically from:

$$\frac{\sinh\delta}{\delta} = \frac{1}{\sqrt{\Delta s_1 \Delta s_2}} \quad (14)$$

Finally, the parameter value corresponding to each point is determined via numerical methods, and following elements are added to the `Side` data structure to preserve these results:

```
struct Side
{       ...
        ...
        int sdnum;
        int sdtype;
        double sdle,sdte;
        double *sdparm;
}
```

where the variable `sdnum` is the number of grid points along the edge, `sdtype` is the mapping type, `sdle` and `sdte` are the end spacing parameters, and `sdparm` is the parameter mapping list.

### Block Interior Grid Construction

Using the parametric point spacing distribution for each block boundary as described above, the interior grid is determined by applying transfinite interpolation to the parameter space of each block. This results in a mesh of parameter values for the block surface from which the grid points are evaluated. The only requirement to determine the interior points is that opposite sides have the same number of points, but not necessarily the same edge point distribution.

## THREE DIMENSIONAL GRID GENERATION

The three-dimensional grid generation method uses the two-dimensional topological block layout as the basic multi-block structure, and a blade model to determine nature of block volumes in the third (i.e., circumferential) dimension.

### Topological layout

Three-dimensional grid generation for turbomachinery applications uses a cylindrical coordinate system fixed to the machine axis as shown in Figure 7. The major assumption for this method is that axisymmetric geometry is used for all engine parts except the blades. This allows all blocks to be created in the cylindrical system as ruled hyperpatches (i.e., in the circumferential direction). The grid is then mapped into the global Cartesian system for display and output.

For the three-dimensional grid generation a general r-z-θ-surface which characterizes the axial and circumferential variation of the flow field is constructed. Then using the two-dimensional multi-block topological definition, the three-dimensional surface is divided into the corresponding blocks, from which tri-parametric hyperpatches are constructed.

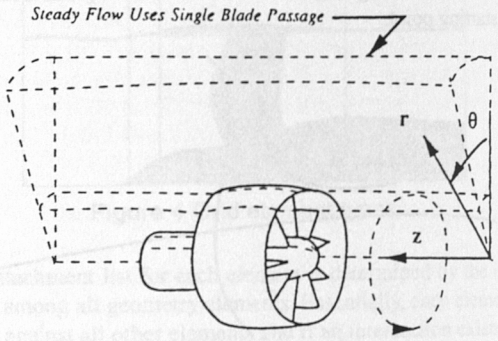


Figure 7 Turbomachinery coordinate system (Crook and Delaney, 1991)

### Axial-Circumferential Surface Construction

The axial-circumferential surface is constructed by extracting iso-parameter curves for each knot value in the blade's span-wise parametric direction. Each iso-parameter curve is extended to characterize the overall shape of the flow field in the axial-circumferential direction. In particular, for each curve, the following information is extracted: leading edge point, trailing edge point, chord length ($C$), inlet angle ($\alpha_i$), and exit angle ($\alpha_e$). From this information a series of five Bezier curves are constructed, as shown in Figure 8. The curve segments upstream of the leading edge are assigned $r$-coordinates equal to that of the leading edge point, and similarly, segments downstream of the trailing edge are assigned $r$-coordinates of the trailing edge point. The maximum and minimum $z$-coordinates are determined from the two-dimensional block structure. The functions $f_\theta(C, \alpha)$ and $f_L(C, \alpha)$ are heuristic linear functions of chord and angle, which are controlled via interactive user input. The Bezier curve segments are then joined into a single *NURBS* curve forming the axial-circumferential flow curve for each span-wise knot of the blade. Two additional curves are formed from axial-circumferential curves corresponding to maximum and minimum knot values. The curves are projected by assigning their r-coordinates to the maximum and minimum values from of the two-dimensional block structure, respectively. Then all the axial-circumferential flow curves are lofted forming the desired flow surface in the cylindrical r-z-θ coordinate system.

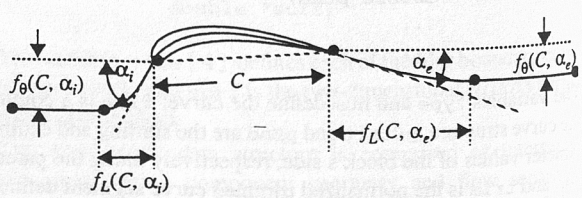


Figure 8 Axial-circumferential curve generation

### Block Generation

The three-dimensional grid generation process is characterized by two types of blocks: non-blade blocks, and blade blocks. A blade block is determined by examining the four corner points of a two-dimensional block. If all the points intersect the blade, the block is a blade block, otherwise, it is a non-blade block. All blade blocks are derived from a single hyperpatch constructed from the pressure and suction sides of the blade, with circumferential variation determined from the number of blades. This hyperpatch characterizes the volume between adjacent blades.

#### Blade Block

Blade blocks are defined on the parameter space of the inter-blade hyperpatch projecting the two-dimensional block structure onto both its pressure and suction sides. This is accomplished by a marching procedure on the block boundaries. The boundaries are traversed, and corresponding blade surface points (neglecting the θ-coordinate) are found via a Newton/Raphson search. The parametric image of the block boundaries on the blade surface is generated by NURBS curve interpolation yielding block definition as a collection of curves in the parameter space of the hyperpatch.

#### Non-Blade Block

Non-blade blocks are generated using a method similar to the blade block construction. A two-dimensional block shape is trimmed out of the r-z-θ surface. Then a ruled hyperpatch is constructed using the trimmed surface and a copy of it offset in the θ-direction by the angular size of the wedge.

### Grid Generation

The following procedure is used to determine the parametric point distribution for each hyperpatch block. The user specifies the edge point distribution on all twelve edges of the hyperpatch via the same mechanism as that applied for two-dimensional grid generation. The only requirement is that the number of edge points must be the same for edges of common spatial orientation, i.e., only one point density may be specified for all *r*-, *z*- and θ-direction edges. Two-dimensional transfinite interpolation is then applied to generate parametric point distributions on each of the six faces of the parametric block. The point distribution on the interior of the hyperpatch is then determined using three-dimensional transfinite interpolation on the six bounding faces. Finally, the grid is constructed by evaluating the hyperpatch at the parameter values and transferring the points from the cylindrical coordinate system to the Cartesian coordinate system.

## EXAMPLES

The following example applications are presented to demonstrate the capabilities and general operation of the two- and three-dimensional grid generation system. Figures 9 shows a typical multi-block structure of an *H*-grid, and Figure 10 shows its associated grid.

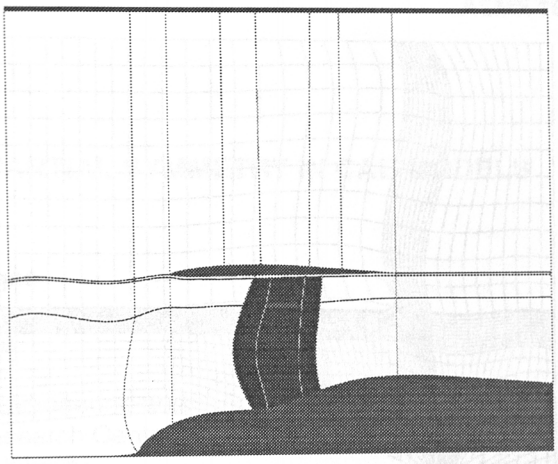

Figure 9 Typical *H*-grid block structure

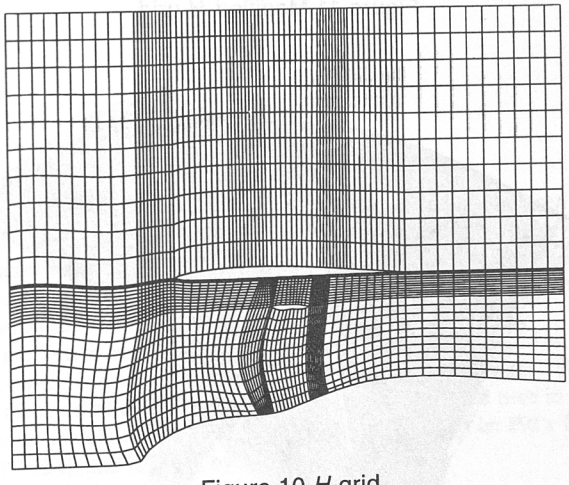

Figure 10 *H*-grid

Figure 11 shows the same grid after several modifications, namely, several blocks above the cowl have been removed, the shape of two blocks upstream of the cowl have been altered, and the grid spacing has been biased toward the hub leading edge.

Figures 12 shows the corresponding three-dimensional grids generated for the same block structure of Figures 9 and 10.

## CONCLUSIONS

By exploiting the strengths of contemporary geometric modeling technology, this new technique provides turbomachinery analysts with a powerful and robust tool for integrated two- and three-dimensional grid generation. The method is implemented within an intuitive graphical user interface which provides complete flexibility for developing and modifying grid structure and point distribution. The IGES interface allows incorporation of existing component models which reduces grid development time, and improves grid accuracy.

Future enhancements include incorporation of capabilities for non-axisymmetric component and flow geometry, non-quadrilateral block definition, and general, three-dimensional *I*-grids.

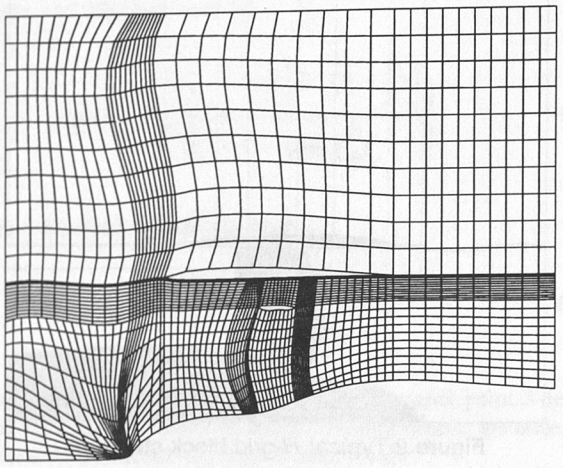

Figure 11 Modified *H*-grid

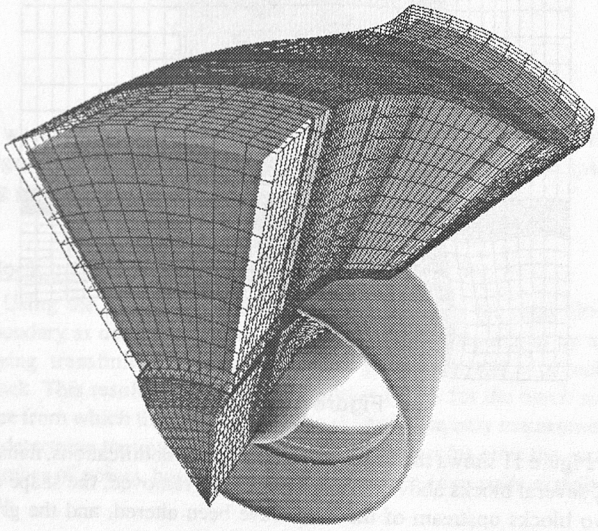

Figure 12 Three-dimensional *H*-grid


## ACKNOWLEGEMENTS

This research was funded by NASA Lewis Research Center under grant number NAG-1318. The authors thank Dr. David P. Miller for his guidance and support.


## REFERENCES


Beach, T.A., 1989, "An Interactive Grid Generation Procedure for Axial and Radial Flow Turbomachinery," NASA CR 185167, NASA Lewis Research Center, Cleveland OH.

Böhm, W., Farin, G. and Kahmann, J., 1984, "A Survey of Curve and Surface Methods in CAGD," *Computer Aided Geometric Design*, Vol. 1, pp. 1-60.

Böhm, W., 1985, "On the efficiency of knot insertion algorithms," *Computer Aided Geometric Design*, Vol. 2, Nos. 1-3, pp. 141-143

Chima, R. V. and Yokota, J. W., 1990, "Numerical Analysis of Three-Dimensional Viscous Internal Flows," *AIAA Journal*, Vol. 28, No. 5, pp. 789-806.

Cox, M., 1972, "The numerical Evaluation of B-splines," *Journal of the Institute of Mathematics Applications*, Vol. 10, pp. 134-149.

Crook, A.J., and Delaney, R.A., 1991, "Investigation of Advanced Counter-rotation Blade Configuration Concepts for High Speed Turboprop Systems, Task III - Advanced Fan Section Grid Generator," NASA Contractor Report CR-187129

Dawes, W. N., 1990, "Towards Improved Throughflow Capability: The Use of 3D Viscous Flow Solvers in a Multistage Environment," ASME Paper No. 90-GT-18, pp. 1-10.

deBoor, C., 1972, "On Calculating with B-splines," *Journal of Approximation Theory*, Vol. 6, pp. 50-62.

Denton, J.D., 1990, "The Calculation of Three Dimensional Viscous Flow Through Multistage Turbomachines," ASME Paper No. 90-GT-19, pp. 1-10.

Farin, G., 1993, *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, San Diego.

Lee, K. D., and Loellbach, J. M., 1989, "A Mapping Technique for Solution-Adaptive Grid Control," AIAA Paper 89-2178

Miller, D.P., 1994, "TIGGERC - Turbomachinery Interactive Grid Generator for Grid Applications and User Guide," to be published, NASA Technical Memo.

Mortenson, M. E., 1985, *Geometric Modeling*, Wiley, New York.

Oliver, J.H., Nair, N.K. and Shanahan, D.E., 1994, "Geometric Design of Turbomachinery Blades on General Stream Surfaces," submitted to *ASME Winter Annual Meeting*, Chicago, IL, November

NIST, 1991, "The Initial Graphics Exchange Specification (IGES) Version 5.1," National Institute of Standards and Technology, Gaithersburg, MD.

Piegl, L., 1991, "On NURBS: A Survey," *IEEE Computer Graphics and Applications*, Vol. 11, No. 1, pp. 55-71

Soni, B., and Shih, M., 1990, "TURBOGRID: Turbomachinery Applications of Grid Generation," AIAA Paper 90-2242, pp. 1-9

Thompson, J. F., Warsi, Z. U. A., and Mastin, C. W., 1985, Numerical Grid Generation, Foundations and Applications, North-Holland, Amsterdam.

Tiller, W., 1992, "Knot-removal algorithms for NURBS curves and surfaces," *Computer Aided Design*, Vol. 8, pp. 445-453

US Navy, 1993, *DT_NURBS Spline Geometry Subprogram Library Users' Manual*, Naval Surface Warfare Center, David Taylor Model Basin, Bethesda, MD